

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр. 8304

Кириянов Д.И.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных функций.

Постановка задачи.

- 1) проанализировать полученное задание, выделив рекурсивно определяемые информационные объекты и (или) действия;
- 2) разработать программу, использующую рекурсию;
- 3) сопоставить рекурсивное решение с итеративным решением задачи;
- 4) сделать вывод о целесообразности и эффективности рекурсивного решения данной задачи.

Вариант 11

Написать программу, которая по заданному (см. предыдущее задание) константному выражению вычисляет его значение либо сообщает о переполнении (превышении заданного значения) в процессе вычислений.

Описание алгоритма.

Для решения поставленной задачи была реализована функция нахождения умножения, которая сразу заменяла умножение на его результат. После чего была реализована рекурсивная функция, которая считает значение выражения и возвращает его. На каждом этапе программа смотрит на знак операции, в зависимости от значения которого вызывается дальнейшая функция. Рекурсия завершается, если дальнейший знак операции не найден.

Спецификация программы.

Программа предназначена для подсчета значения выражения. Программа написана на языке C++. Входными данными является строка — математическое выражение. На выход программа дает

решение выражения, либо сообщает о переполнении.

Описание функций.

Функция `void mult(char* arr)` – находит умножение в строке при помощи `strstr` и далее заменяет его на значение при помощи `atoi` и `memmove`. Принимает адрес строки выражения.

Функция `int expression(char* mas, int result, char* flag)` – рекурсивная функция, которая считает значения выражения. Принимает адрес строки выражения, текущее значение выражения и флаг, служащий для проверки на переполнение. Завершается, если знак операции не найден.

Вывод.

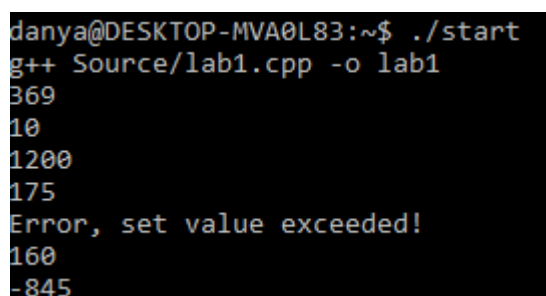
В ходе работы были изучены рекурсивные методы решения задач, был получен опыт работы с рекурсией. На мой взгляд данную задачу можно было реализовать без использования рекурсии.

ПРИЛОЖЕНИЕ

1) ТЕСТИРОВАНИЕ:

Работа программы, входные даны в таблице

| | |
|--|----------------------------|
| 123*3 | 369 |
| 5+5 | 10 |
| 100+100+200*5 | 1200 |
| 200-25 | 175 |
| 999999999+999999999+99999999+99999999 99+999999999999 | Error, set value exceeded! |
| 15+15*5-10*3+100 | 160 |
| 20-100-125*3+3-20*20+7 | -845 |



```
danya@DESKTOP-MVA0L83:~$ ./start
g++ Source/lab1.cpp -o lab1
369
10
1200
175
Error, set value exceeded!
160
-845
```

2) ИСХОДНЫЙ КОД:

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <string>
#include <cstring>
#include <climits>
int expression(char* mas, int result, char* flag) { //рекурсивная функция
    while (isdigit(*mas))
        mas++;
    if (*mas == '+') {
        mas++;
        if (result < INT_MAX - atoi(mas)) { //если нет переполнения, рекурсия продолжается
            result = expression(mas, result + atoi(mas), flag);
        }
        else {
            *flag=1; //если переполнение, то ставим флаг
        }
    }
    else if (*mas == '-') {
        mas++;
        if (result > INT_MIN + atoi(mas)) { //аналогично для отрицательных значений
            result = expression(mas, result - atoi(mas), flag);
        }
        else {

```

```

        *flag=1;
    }
}
return result;//функция возвращает результат выражения
}

void mult(char* arr) {
    char* search = strstr(arr, "*");//ищем знак умножения
    while (search != NULL) { //заменяем умножение на результат
        search -= 1;
        int a = 1;
        while (isdigit(*(search)) && (search > arr)) {
            search -= 1;
            a += 1;
        }
        if (search != arr) {
            search += 1;
            a -= 1;
        }
        int b = (atoi(search) * atoi(search + a + 1));
        char beta[80];
        sprintf(beta, "%d", b);
        search = search + a + 1;
        int c = 0;
        while (isdigit(*(search))) {
            c++;
            search++;
        }
        char* next = search;
        search = search - a - c - 1;
        memmove(search, beta, strlen(beta));
        search = search + strlen(beta);
        memmove(search, next, strlen(next) + c);
        search = strstr(search, "*");
    }
}

int main(int argc, char* argv[]){
    int memory = 50;
    char* arr = (char*)malloc(memory * sizeof(char)); //выделяем память для массива
    char* flag = (char*)calloc(3, sizeof(char)); //флаг для рекурсивной функции
    if (argc == 1) { //считывание с консоли
        int index = 0;
        char s;
        for (; ; ) {
            s = getchar();
            if (s == '\n')
                break;
            arr[index] = s;
            index++;
            if (index == memory - 1) {
                memory += 50;
                arr = (char*)realloc(arr, memory * sizeof(char)); //перевыделение
                памяти, если требуется
            }
        }
        arr[index] = '\0';
        mult(arr); //вызов функции поиска умножения
        int res = expression(arr, atoi(arr), flag); //вызов рекурсивной функции подсчета
        выражения
        if (*flag!=1)
            std::cout << res << '\n'; //если не было переполнения, выводим результат
        else
            std::cout << "Error, set value exceeded!\n"; //иначе выводим сообщение о
        переполнении
    }
}

```

```

    }
    else { //считывание из файла
        FILE* file = fopen(argv[1], "r"); //открываем файл
        if (!file) { //если файл не открыт выводим сообщение и завершаем программу
            std::cout << "Incorrect file name\n";
            return 0;
        }
        int index = 0;
        char c = getc(file);
        while (c != EOF) {
            while ((c != '\n') && (c != EOF)) {
                arr[index] = c;
                index++;
                if (index == memory - 1) {
                    memory += 50;
                    arr = (char*)realloc(arr, memory *
sizeof(char)); //перевыделение памяти, если требуется
                }
                c = getc(file);
            }
            arr[index] = '\0';
            mult(arr); //вызов функции поиска умножения
            int res = expression(arr, atoi(arr), flag); //вызов рекурсивной функции
подсчета выражения
            if (*flag != 1)
                std::cout << res << '\n'; //если не было переполнения, выводим
результат
            else
                std::cout << "Error, set value exceeded!\n"; //иначе выводим сообщение
о переполнении
            *flag = 0; //обнуляем флаг после каждого цикла
            index = 0; //обнуляем индекс после каждого цикла
            c = getc(file);
        }
        fclose(file);
    }
    free(flag);
    free(arr);
    return 0;
}

```